Reconfigurability in MDO Problem Synthesis, Part 1

AIAA – 2004-4307

Natalia M. Alexandrov and Robert Michael Lewis

NASA Langley Research Center
Hampton, Virginia

College of William & Mary
Williamsburg, Virginia

http://mdob.larc.nasa.gov

10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference

Outline

Companion papers: Part1 – approach; Part 2 – details

- Focus: MDO-NLP
- Idea of reconfigurable MD synthesis (REMS)
- Basic tools of REMS
- REMS process
- Relation to other efforts
- Concluding remarks

MDO-NLP

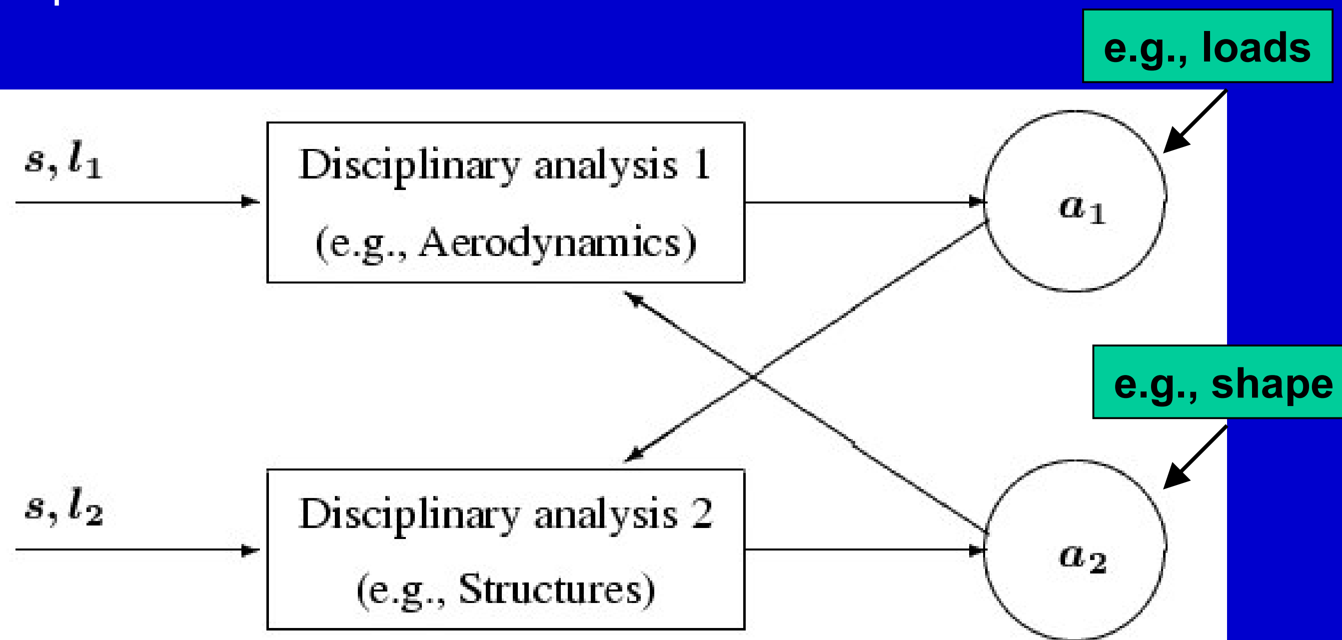
- MDO ≡ part of the total design process that can be stated as a nonlinear programming problem (NLP) (our focus to-date)
- MDO-NLP formulation
  - Influences computational tractability of optimization problem
  - In realistic problems, formulation may not be clear *a priori*
- Observing MDO application teams
  - Before optimization is considered, significant time and effort spent on developing multidisciplinary analysis (MDA)
  - Little or no room for experimentation with alternatives
- Experience with MDO test problems
  - Always in the form of a monolithic formulation
  - Disciplinary components hidden in implementation
  - “Dis-integrating” problems to experiment with alternative formulations is time-consuming and error prone

MDO-NLP

- Extensive work devoted to MDO-NLP decomposition
- “Decomposition” assumes an ur-problem
- Our perspective
  - There is no ur-problem: MDO starts out as a collection of autonomous disciplinary analyses with diverse data formats
  - The task is to assemble an MDO formulation from autonomous disciplinary information
  - Make it as easy as possible on all concerned
- Clear need
  - Flexible MDO problem abstraction to assist researchers and practitioners in formulating and re-formulating MDO problems with maximum possible ease
  - I.e., need a language for reasoning about MDO

Idea of reconfigurable MD synthesis (REMS)

- Capacity for reconfigurability among MFO formulations:
  - sharing basic computational components
  - being related via closer of analysis constraints and variable eliminations
- Two-discipline model problem:



- Coupled MDA $\sim$ the physical requirement that a solution satisfy both analyses
- Given $x = (s, l_1, l_2)$, we have

$$
\begin{aligned}
a_1 &= A_1(s, l_1, a_2) \\
a_2 &= A_2(s, l_2, a_1)
\end{aligned}
$$

Simultaneous Analysis and Design (SAND)

Relax all couplings;
All variables independent

Write MDA as
$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) \\ a_2 &= A_2(s, l_2, t_1) \\ t_1 &= a_1 \\ t_2 &= a_2 \end{aligned}$$

$$\underset{s, l_1, l_2, a_1, a_2, t_1, t_2}{\text{minimize}} \quad f(s, t_1, t_2)$$

subject to

disciplinary constraints
$$\begin{cases} c_1(s, l_1, a_1) \geq 0 \\ c_2(s, l_2, a_2) \geq 0 \end{cases}$$

analysis constraints
$$\begin{cases} a_1 = A_1(s, l_1, t_2) \\ a_2 = A_2(s, l_2, t_1) \end{cases}$$

consistency constraints
$$\begin{cases} t_1 = a_1 \\ t_2 = a_2 \end{cases}$$

Distributed Analysis Optimization (DAO)

Close disciplinary consistency constraints;

relax the coupling in MDA; maintain disciplinary analyses

A DAO formulation is

$$
\begin{array}{ll}
\underset{s, l_1, l_2, t_1, t_2}{\text{minimize}} & f(s, t_1, t_2) \\
\text{subject to} & \left.\begin{array}{l} c_1(s, l_1, t_1) \geq 0 \\ c_2(s, l_2, t_2) \geq 0 \end{array}\right\} \text{disciplinary constraints} \\
\text{consistency constraints} & \left\{\begin{array}{l} t_1 = a_1(s, l_1, t_2) \\ t_2 = a_2(s, l_2, t_1), \end{array}\right.
\end{array}
$$

where the disciplinary responses $a_1(s, l_1, t_2)$ and $a_2(s, l_2, t_1)$ are found by closing the disciplinary analysis constraints

$$
\begin{aligned}
a_1 &= A_1(s, l_1, t_2) \\
a_2 &= A_2(s, l_2, t_1).
\end{aligned}
$$

(AKA Individual Discipline Feasible, Cramer et al.)

Fully Integrated Optimization (FIO)

Close multidisciplinary consistency constraints

The corresponding FIO formulation is

$$\begin{array}{ll} \underset{s,l_1,l_2}{\text{minimize}} & f(s, t_1(s, l_1, l_2), t_2(s, l_1, l_2)) \\ \text{subject to} & c_1(s, l_1, t_1(s, l_1, l_2)) \geq 0 \\ & c_2(s, l_2, t_2(s, l_1, l_2)) \geq 0 \end{array}$$

where we compute $t_1(s, l_1, l_2)$ and $t_2(s, l_1, l_2)$ by solving the MDA

$$\begin{array}{lllllll} a_1 & = & A_1(s, l_1, t_2) & \qquad & t_1 & = & a_1 \\ a_2 & = & A_2(s, l_2, t_1) & \qquad & t_2 & = & a_2. \end{array}$$

Formulations and reconfigurability, cont.

- Eliminating (a_1,a_2) via disciplinary analyses + eliminating (l_1,l_2) via disciplinary design constraints generally leads to bilevel optimization problems
- Minimal computational components can be re-used
- Standard results on reduced derivatives tell us that the sensitivities in DAO and FIO are related to those in SAND *via variable reduction*
- Therefore, computational components of one formulation can be reconfigured to yield those of another in the context specific algorithms
- For a specific choice of algorithm (e.g., reduced-basis SQP) and specific formulations (e.g., DAO, FIO, SAND), the relationship among the sensitivities means that it is possible to implement an optimization algorithm for SAND so that with a single modification one obtains an algorithm for DAO or FIO (Lewis 1997)

Role of abstraction

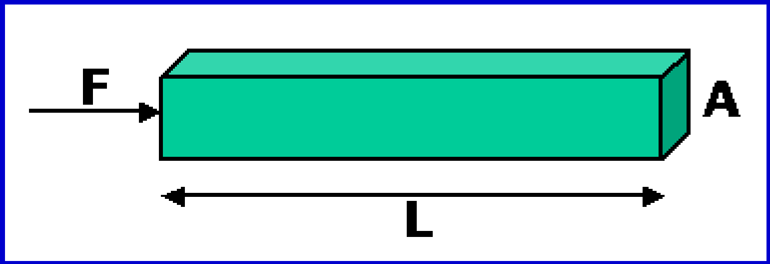
- Reasoning about MDO (NLP) formulation involves problem specification or notation
- Algebraic specification is well suited for NLP problem formulation
  - Example: AMPL (A Modeling Language for Mathematical Programming, Fourer *et al*.)
  - In fact, we would like to come up with AMPL for MDO
- From a user's perspective
  - Algebraic specification for MDO is difficult for more than two disciplines: need to distinguish among variables shared by several pairs of disciplines; may be duplicates
  - Would like to have problem specification in a subset of a natural language (English) and handle the assembly as automatically as possible

Components of REMS

- Problem specification
  - Lists of disciplinary inputs and outputs of the form
    Identifier Description Attributes
- Abstraction – directed graphs representing data flow
  - Function nodes
    - Disciplinary or subsystem operations
    - Objectives and constraints
    - May contain hierarchies or simple operations
  - Data nodes
    - Inputs and outputs of functions
    - A single output may serve as input to several functions
- Basic approach – compiler-like assembly and manipulation of information from nodes

REMS process illustrated with a simple example

- Two "disciplines", stress $\mathcal{S}$ and weight $\mathcal{W}$, govern the behavior of a bar under a load F



Step 1: Autonomous disciplinary description

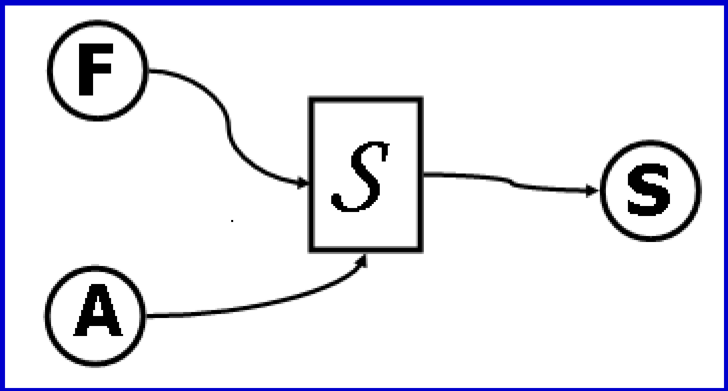
- Disciplinary practitioners describe inputs and outputs of $\mathcal{W}$ and $\mathcal{S}$, autonomously, without reference to multidisciplinary context:

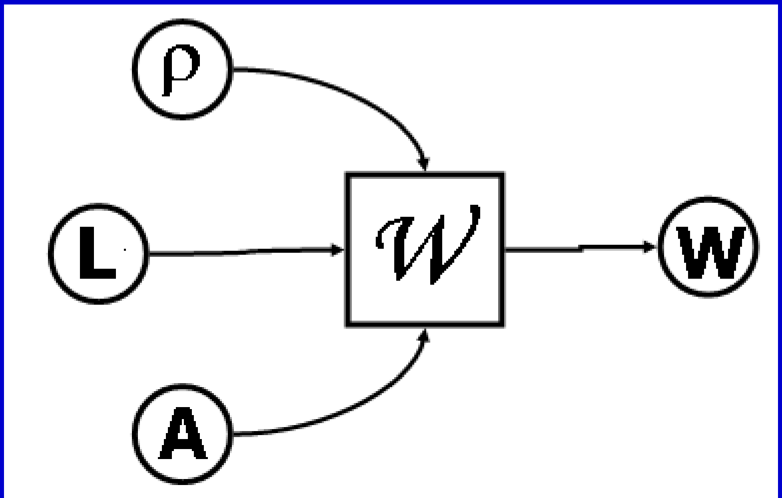
ID	Description	O or I	Dimension	…
A	cross-sectional area	I	1	
F	longitudinal stress	I	1	
S	stress	O	1	

REMS process, step 1, cont.

- Similarly, for the other “discipline”

A	cross-sectional area	I	1	…
ρ	density	I	1	
L	length	I	1	
W	weight	O	1	





REMS process, step 1, cont.

- Autonomous disciplinary specification of inputs and outputs – a simple task than accounting for I/O in multidisciplinary context at the outset
- Problem representation remain dynamic throughout formulation process
  - Need not describe all data
  - Need not have an exhaustive list of attributes

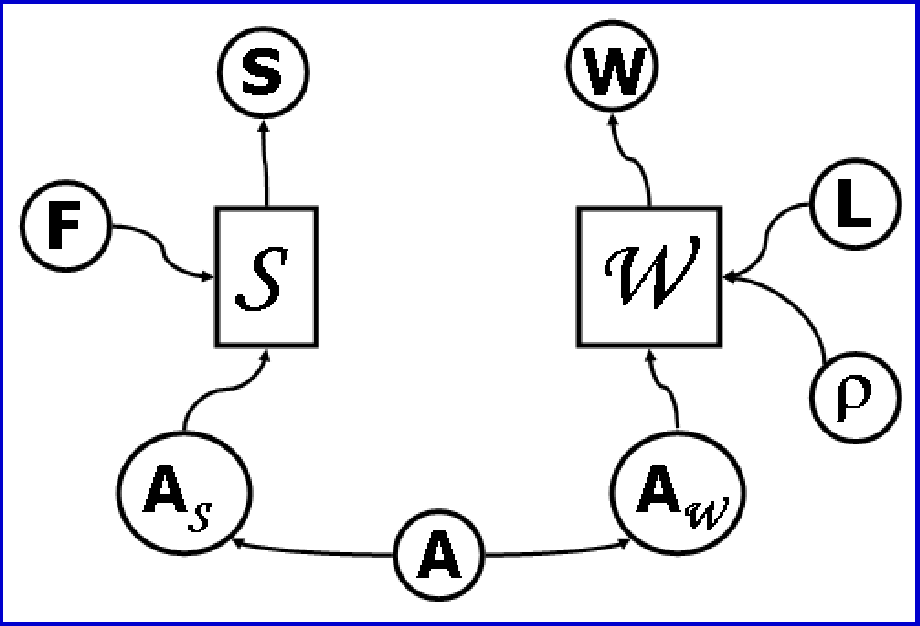
REMS process, step 2: compiling disciplinary IR

- REMS examines the disciplinary I/O lists and automatically assembles intermediate representations of subsystems (disciplines) as function nodes with in and outgoing data nodes
- Incidence matrices are constructed (all nodes vs. all nodes, with 1 or 0 entries in the matrix)
- At this stage REMS can compile disciplinary sensitivity information

REMS process, step 3: reconciling MD coupling

- Link disciplinary IR into a multidisciplinary IR
- Detects opportunities for distributed computation
- Opportunity to check for coupling bandwidth
- Opportunity to check for errors and intentions of practitioners
  - E.g., A function node expects an input but does not have one with an expected identifier

- In realistic applications expects disciplinary experts to communicate at this stage
- Can help compile data dictionaries or thesauri
- Can use data dictionaries to decrease interaction at this stage



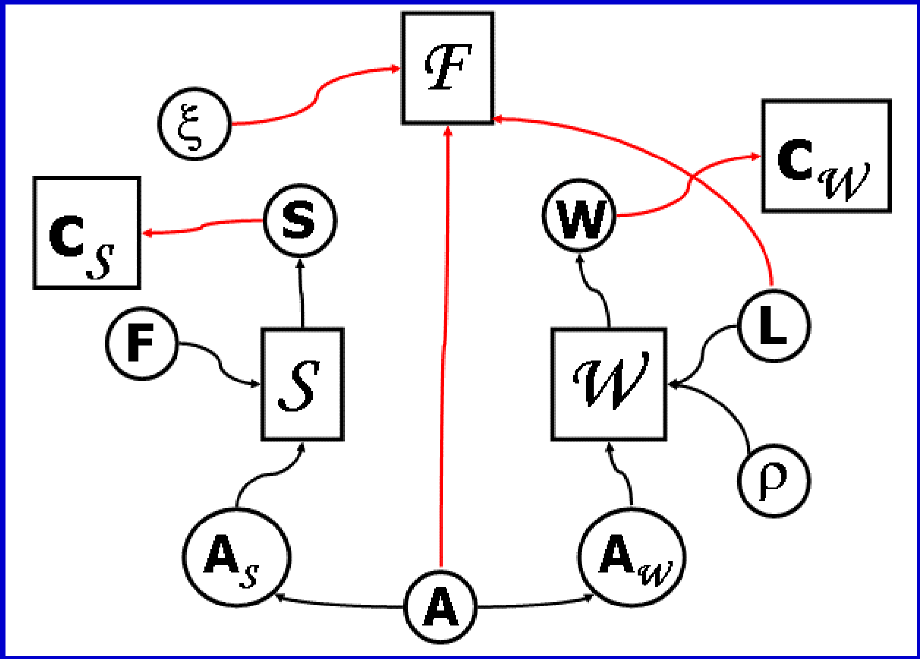
REMS process, step 4: objective and constraint identification

- Identify objective and constraint information
- Leaf nodes are all potential objectives and constraints
- Examine problem formulation
- Assemble conceptual sensitivity information for optimization formulation

$$\underset{A}{\text{minimize}}\ \mathcal{F} = \xi\, L\, A$$

$$\text{subject to } S = F/A \leq S^*$$

$$W = \rho\, L\, A \leq W^*$$



Summary of the process

- Start with disciplinary data description
- Translate description into intermediate representations
- Link intermediate representations and generate incidence matrix
- Continue with the iteration
  - Analysis of intermediate representation
  - Manipulation of representations
  - Updates
- N.B. So far, avoided difficulties with algebraic notation

Summary of the process, cont.

- Tasks
  - Error checking
  - Derivative composition
  - Propagation of local problem changes throughout formulation
  - In highly structured contexts, manipulation of sensitivity information to be passed among various formulations

REMS in relation to other methods

- Many connections with other efforts
  - Computational components pervasive in scientific computation; e.g., AMPL (Fourer *et al*.), TAO (Benson *et al*.)
  - Using graph abstractions to examine decompositions (Wagner)
  - Using abstract language (χ) to coordinate design process (Etman *et al*.)
  - Computational frameworks (e.g., ModelCenter, DAKOTA) must rely on abstractions of computational components
- The goals of REMS are complementary
  - To our knowledge, most efforts start with a conceptual NLP formulation and make decisions about decomposition and coordination
  - Our goal is to start reasoning about the problem *before* it is conceptually formulated or integrated into a framework
  - View REMS as a potential pre-processor in frameworks

Concluding remarks

- Logical framework for MDO problem specification and reasoning
- Applicable to other problems of similar structure in the context of NLP (e.g., synthesis of large single-discipline problems following domain decomposition)
- General ideas are likely applicable to reasoning about complex systems in broader contexts (e.g., systems of systems)
- A grammar defined
- Language and automatic analysis and manipulation of representations under development